Things I've Noticed Along the Way

Dr. Robert Colwell

Salishan Conference on High-Speed Computing

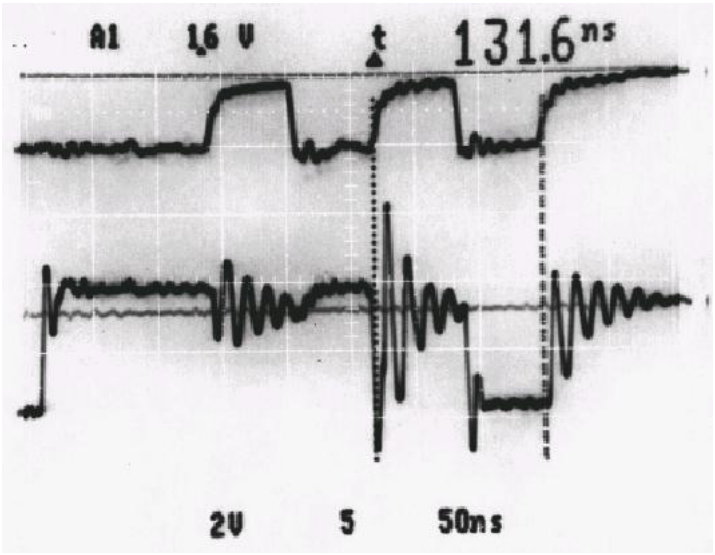
April 2013

What They Didn't Teach In EE Classes: Ground Bounce

TTL industry melted down in 1980's with advanced Schottky

$V = L * dI/dT$

If L is non-trivial, current I is large, and dT is short (fast edges), badness ensues…



Children know about ground bounce – they can *feel* it

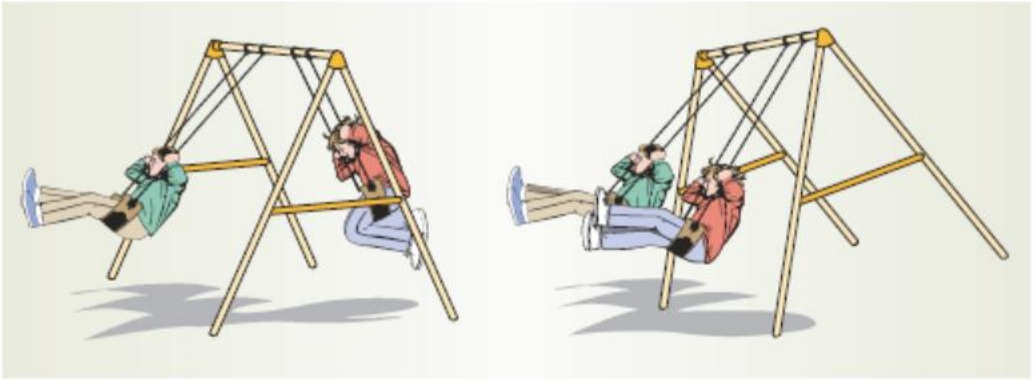


Figure 1. (left) Swinging out of phase demonstrates how out-of-phase outputs cancel, and there is no ground bounce. (right) Swinging in the same direction demonstrates in-phase outputs, adding up to ground bounce.

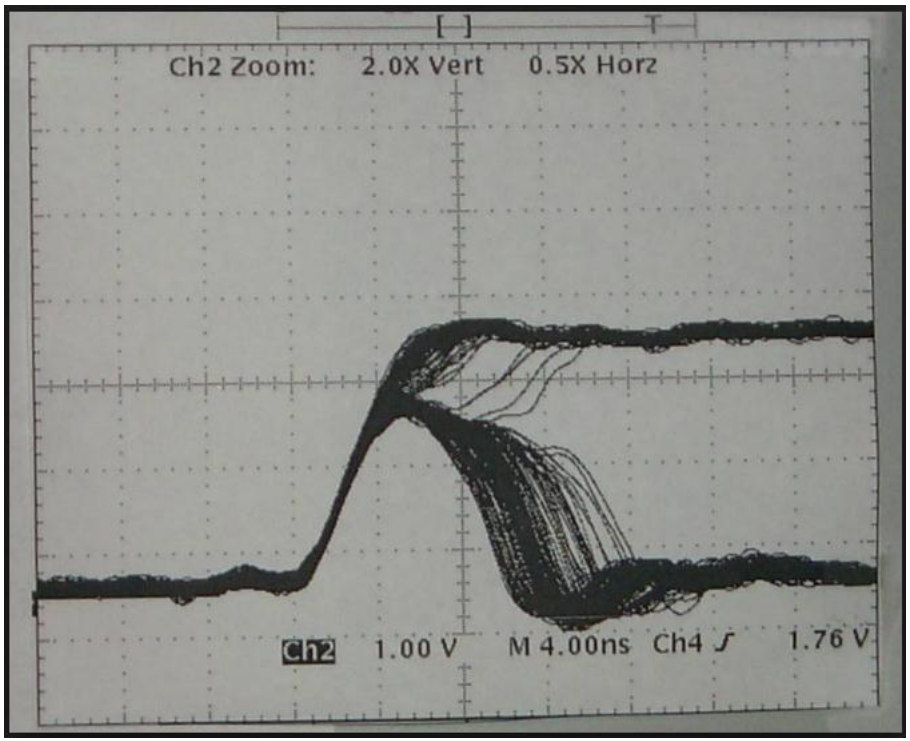
What They Didn't Teach In EE Classes: Metastability

Metastability: built into nature

First widely publicized in 1960's by Chaney & Molnar

All synchronous systems with an async interface have probabilistic I/O's

Still a surprise to us on Bellmac-32



> Suppose two similar dates in front of a man, who has a strong desire for them but who is unable to take them both. Surely he will take one of them, through a quality in him, the nature of which is to differentiate between two similar things.
>
> — *Abu Hamid al-Ghazali, The Incoherence of the Philosophers 1100*[3]

There is no digital. There is only analog, and probabilities.

If that doesn't scare you a little, then think about it some more.

What They Don't Even Know In SW Classes

Software is where complexity goes to cause maximum schedule slips

- Tools and methods can be better or worse, but basic problem is sheer complexity of system design, and fact that it all gets stuffed into the software.
- F22, F35, and Chevy Volt
- We are not one good programming language away from solving this problem

Lead Designers must be able to say "NO!" and make it stick.

(see: Kelly Johnson & SR-71: "More Than My Share of It All")

If nobody does this function, then the project will continuously absorb changes, miss the deadline, absorb more changes, and end up a muddled mess that satisfies no one.

What They Don't Even Know In SW Classes

Software is where complexity goes to cause maximum schedule slips

> Bogdan cited three other concerns with the F-35 program that he said may cause delays. Calling the plane "a flying software computer," he said that software development is behind schedule and may slip further behind as its complexity increases.
>
> "Software is a huge risk, and we've got to do business a little differently," he said.
>
> Bloomberg.com 9/17/2012

But the "gorilla in the room," General Bogdan said, is testing and securing the 24 million lines of software code for the plane and its support systems, a mountain of instructions that goes far beyond what has been tried in any plane.

Do They Even *Teach* Validation at Universities? (Would it help?)

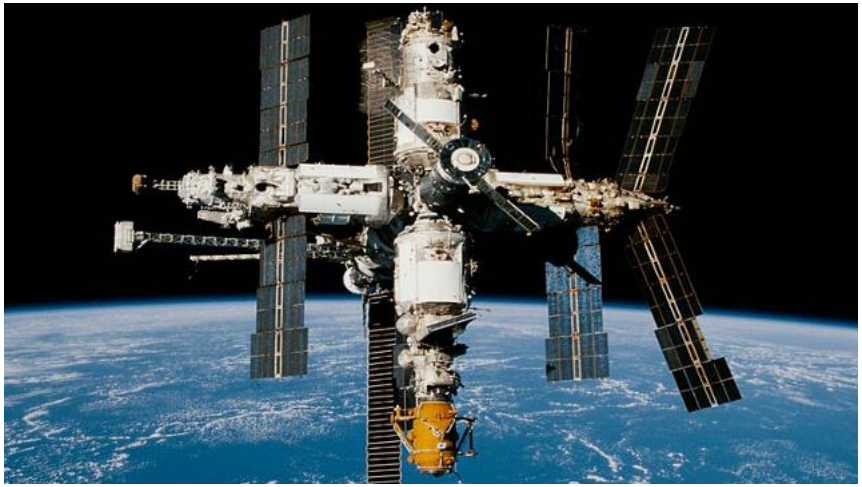
Validation/verification is the most thankless job

- You cannot test to saturation, so there will always be a nonzero chance of escape.
- Validation is on the critical path. It is not scaling.
- If a bug escapes into the wild, they blame you, even though you didn't create the bug.
- If you get 'em all, they think you're overstaffed and too expensive.
- Dangers of always starting simulations at same place (reset)
- Dangers of formal methods (FDIV)
- Murphy's Law is misunderstood…things do NOT always go as badly wrong as they possibly can (near misses are even more important)

If you didn't test it, it doesn't work.

Validation Lesson: Fire Aboard MIR Space Station

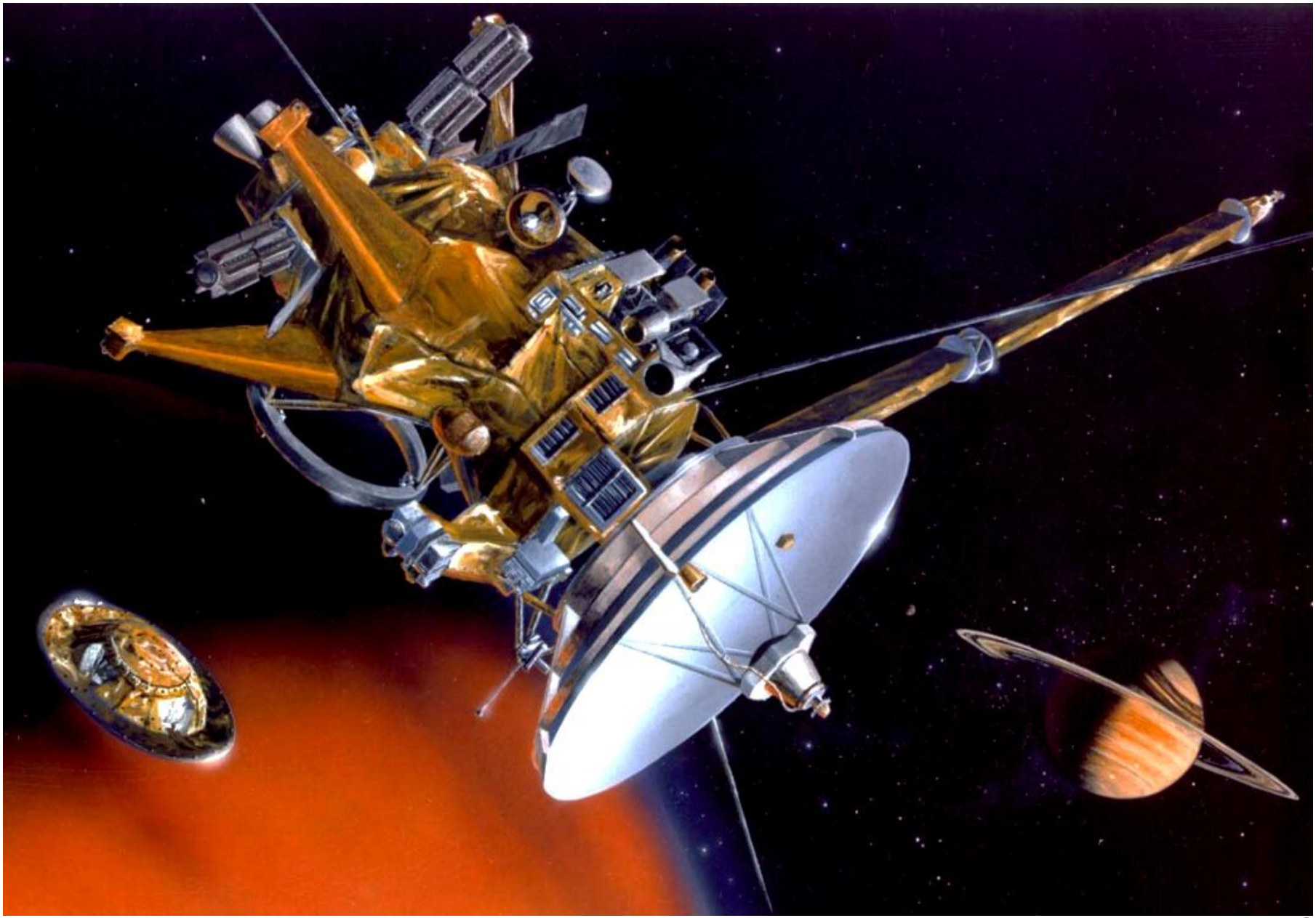
- February 1997…U.S. Astronaut Jerry Linenger on Russian MIR Space Station
- Two 3-person teams on board
- Oxygen generator caught fire
  - worst on-board threat in a spacecraft



- Emergency drills for fire never practiced
- Fire blocked only path to one of the escape capsules
- Extinguishers were still bolted to wall in launch config, needed tools
- Linenger concluded
  - "emergency procedures need to be debated, thought out…"
  - Extinguisher design itself precluded full-up tests, because pulling them from wall activated them for 3 months, then they'd need replacement
  - Had both capsules been used, they'd have collided during re-entry (same coords)
- I conclude
  - Needed to do full-up tests to realize escape path problem & exting. mount issue

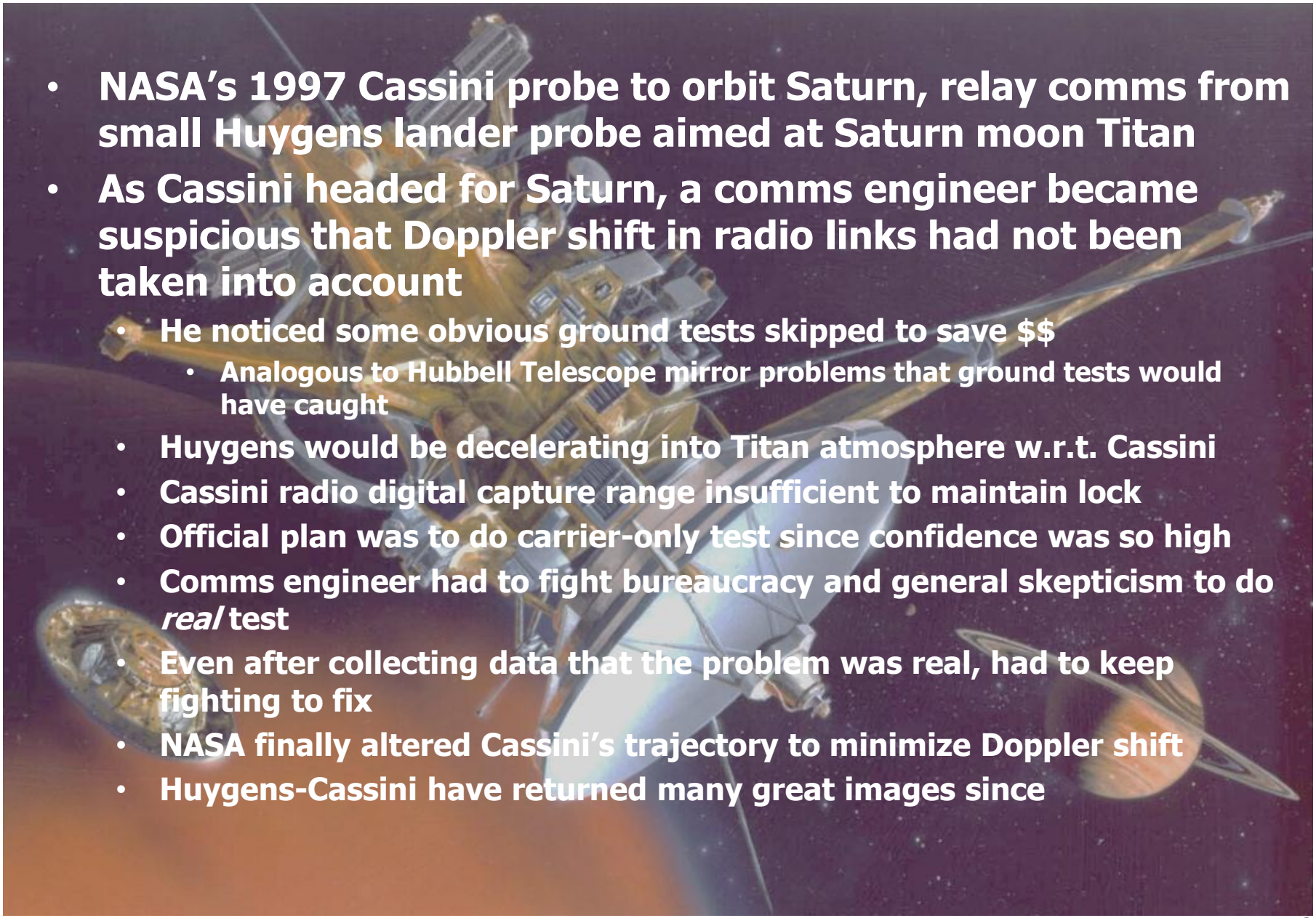
NASA System Failure Case Studies, Trial By Fire, Nov. 2011 V5 Issue 11

Validation Lesson: nearly lost lander due to Doppler shift



http://spectrum.ieee.org/aerospace/space-flight/titan-calling/0

Validation Lesson: nearly lost lander due to Doppler shift



- **NASA's 1997 Cassini probe to orbit Saturn, relay comms from small Huygens lander probe aimed at Saturn moon Titan**
- **As Cassini headed for Saturn, a comms engineer became suspicious that Doppler shift in radio links had not been taken into account**
  - **He noticed some obvious ground tests skipped to save $$**
    - **Analogous to Hubbell Telescope mirror problems that ground tests would have caught**
  - **Huygens would be decelerating into Titan atmosphere w.r.t. Cassini**
  - **Cassini radio digital capture range insufficient to maintain lock**
  - **Official plan was to do carrier-only test since confidence was so high**
  - **Comms engineer had to fight bureaucracy and general skepticism to do *real* test**
  - **Even after collecting data that the problem was real, had to keep fighting to fix**
  - **NASA finally altered Cassini's trajectory to minimize Doppler shift**
  - **Huygens-Cassini have returned many great images since**

http://spectrum.ieee.org/aerospace/space-flight/titan-calling/0

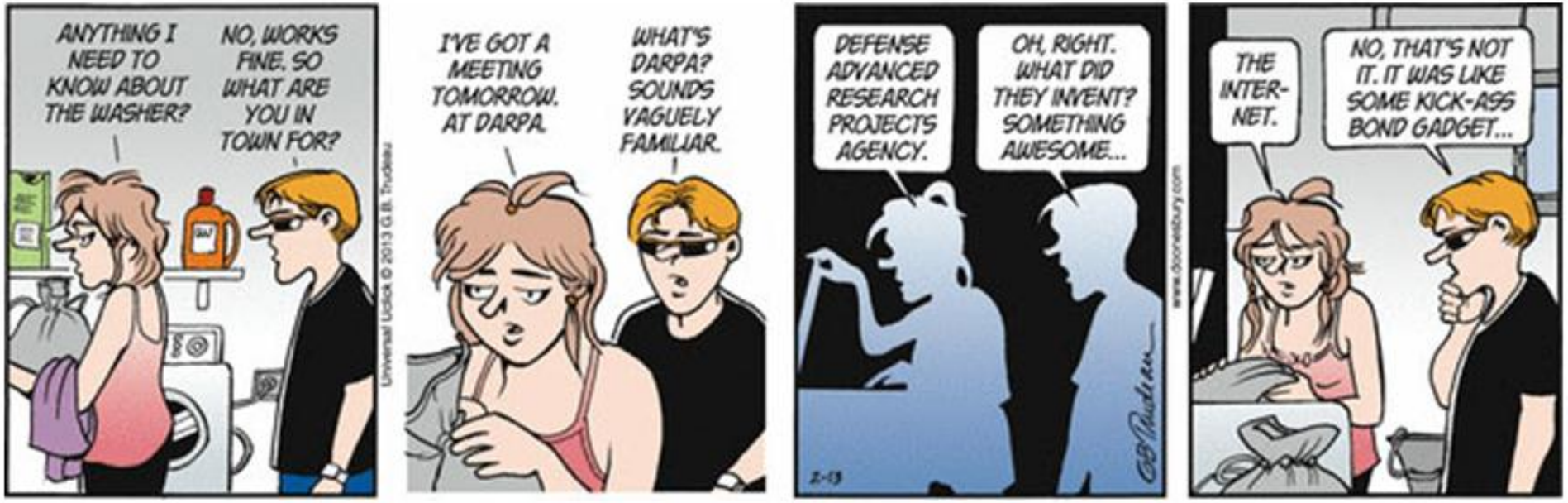
How Not To Design Things

Bell Labs

- Pecking order trumps design on mediocre teams
- PLAID debacle

Multiflow follies

- Flaming fireballs
- Hide the GC incident
- nDot initialization incident
- 300 blue wires
- “latent design faults” paper and crowd reaction



GARRY TRUDEAU